Angular

Universal

Expliqué par David Gilson

https://github.com/gilsdav

Assets

- https://github.com/gilsdav/aa-universal-workshop
- Angular CLI 7

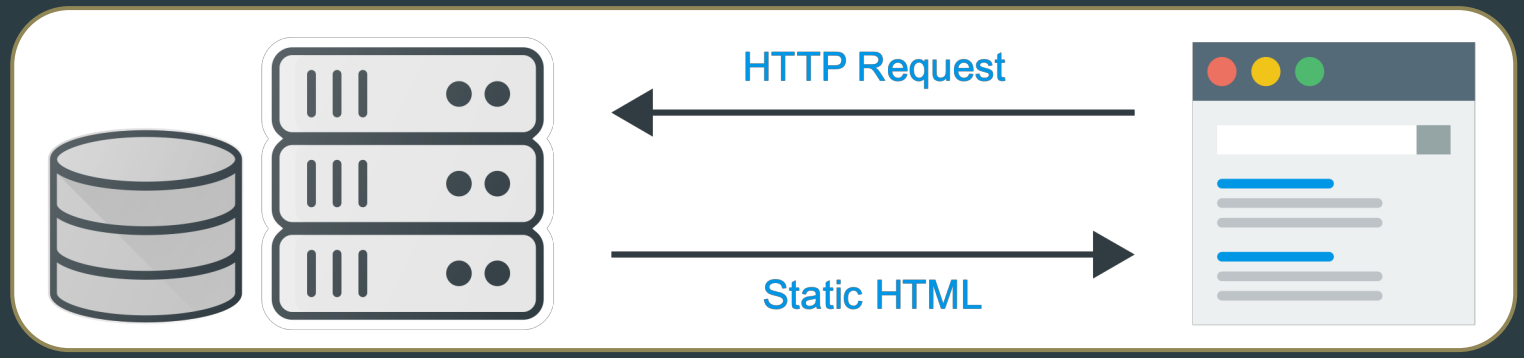
Sommaire

- Qu’est-ce que Angular Universal ?
  - SSR vs CSR
- Quand l’utiliser ?
- Pourquoi l’utiliser ?
- Fonctionnement interne
- Implémentation
  - Ajout d’Universal dans un projet
  - Éviter les doubles appels API
- Universal avancé
  - Page 404
  - Cache serveur
  - SEO

Qu’est ce que Angular Universal ?

- Angular Universal est une technologie permettant de créer un rendu statique d’une application Angular côté serveur (Server Side Rendering).

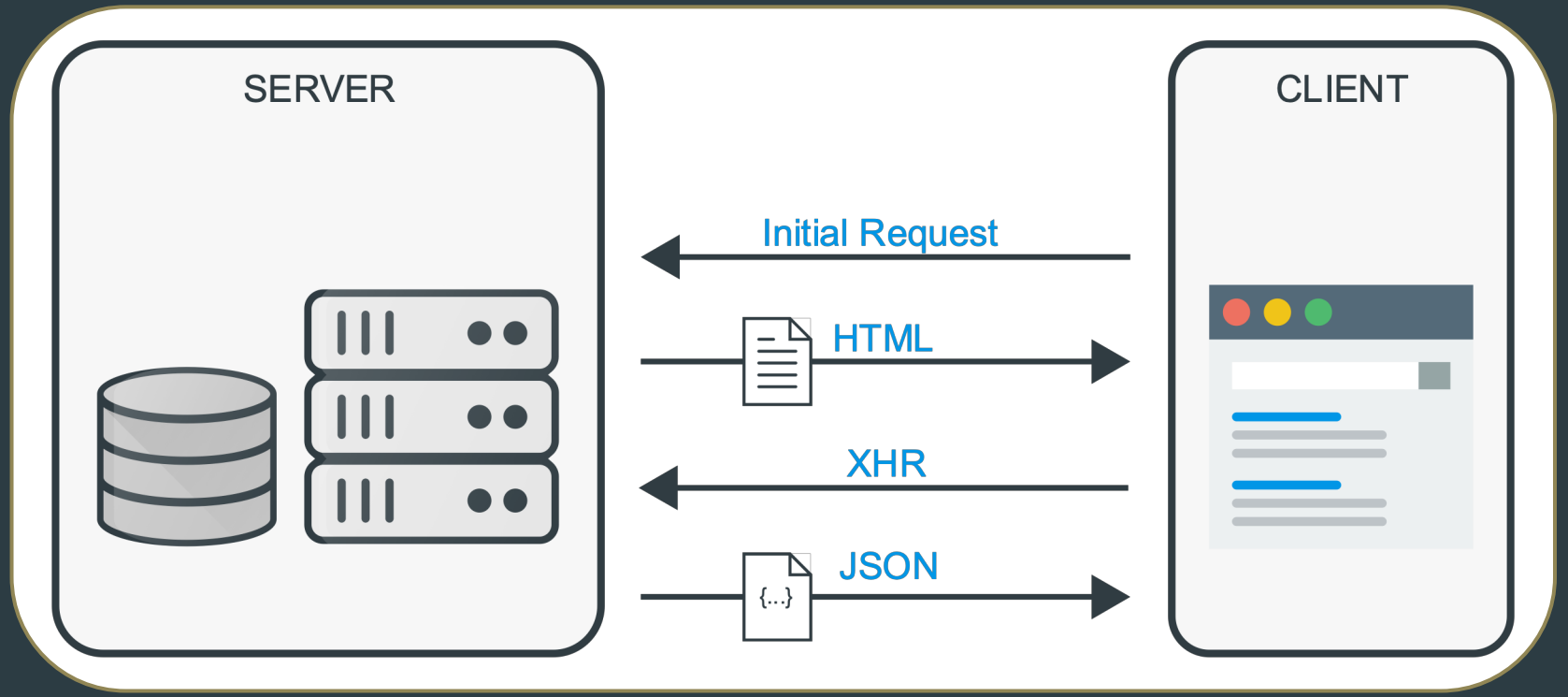
- SSR est l’ « ancienne » façon de rendre des sites web (JSF, Razor, PHP…)



Avantages/Inconvénients du SSR

- Avantages
  - Search Engine Optimization (SEO)
  - Metas de « preview » lisibles par les réseaux sociaux (exemple: OG)
  - Adapté aux sites de contenu statique (exemple: un blog)
    - Mise en cache possible
- Inconvénients
  - Nécessite le rafraichissement complet de la page pour afficher du nouveau contenu
  - Demande de ressource serveur plus élevée

CSR (SPA)



Avantages/Inconvénients d'une SPA

- Avantages
  - Expérience utilisateur complètement interactive
  - Application utilisable en cas d'erreur serveur (API)
- Inconvénients
  - Néant au niveau du SEO
  - Chargement de l'application plus long

Rendu SPA

```
<html>
   <head> ... </head>
   <body>
      <app-root>Loading...</app-root>
      <script src="myapp.js"></script>
   </body>
</html>
```

Rendu SSR

```
<html>
  <head> ... </head>
  <body>
    <app-root>
      <a routerLink="/" href="http://localhost:4200"></a>
      <router-outlet></router-outlet>
      <my-component>
        <h1>Hello World</h1>
      </my-component>
    </app-root>
    <script src="myapp.js"></script>
  </body>
</html>
```

Que choisir ?

- Ne pouvons-nous pas avoir le meilleur des deux mondes ?
- CSR + SSR =

Universal – le meilleur des deux mondes

- Angular Universal a été ajouté dans les **repos officiels** à partir de la version **5** d'Angular.

Universal – Quand ?

- Adapté à
  - Site de contenu « statique »
    - même contenu pour tous les clients
      - Presse, Blog, Site vitrine
- Non adapté à
  - Application complètement dynamique
    - Qui change à chaque rafraichissement de page
  - Application contextualisée par une authentification
    - Tout ce qui se trouve derrière une authentification ne pourra être visible par le SSR
  - Intranet

Universal – Pourquoi ?

- Les gains pour votre SPA
  - SEO (crawl-able)
  - Link Preview (scrape-able)
  - Temps de chargement initial

Problématique SEO

- Pourquoi les Search Engine n'arrivent-ils pas bien à indexer les SPA ?
  - Parce qu'il doit deviner quand la page est complètement chargée.

Problématique Link Preview

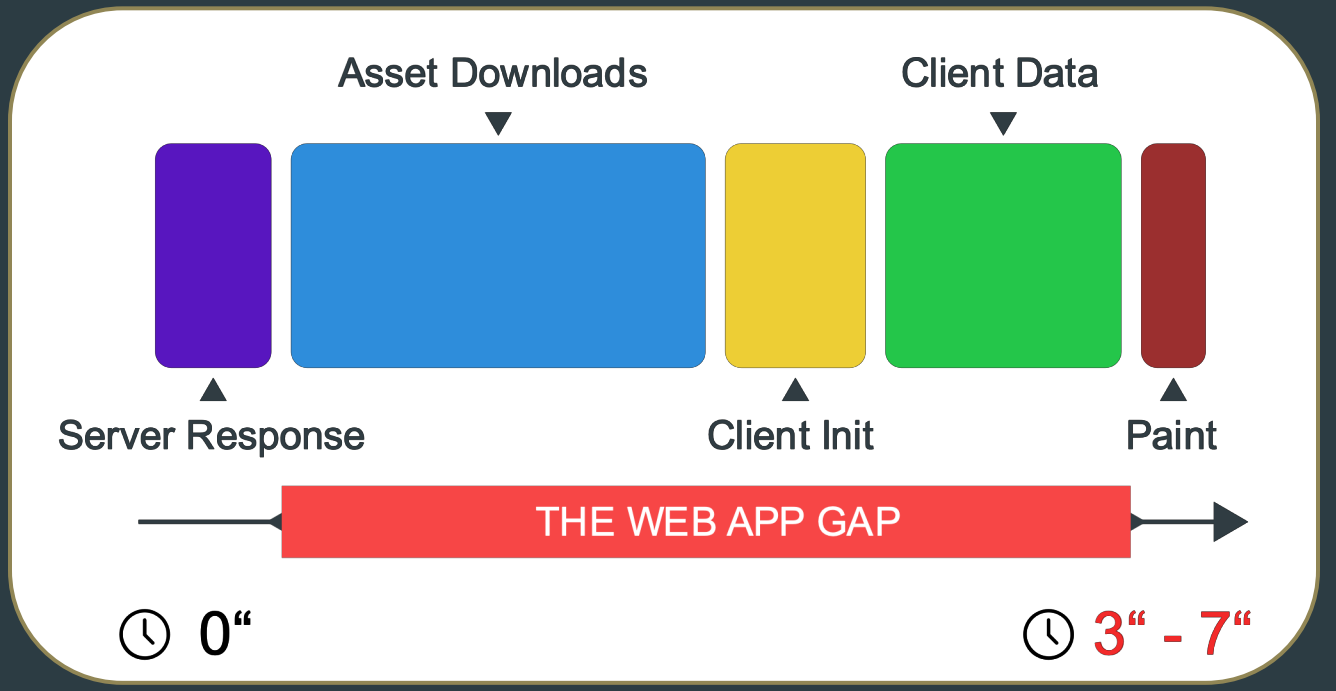
- Les applications utilisant les métas, tel que OG, ne prennent en compte que le code source de la page.
  - Aucun JavaScript ou CSS n’est pris en compte

Problématique Temps de chargement

- Angular fournit-il, de base, des outils pour réduire le temps de chargement ?
  - AOT
  - Lazy-loading

- Le temps qu'il faut pour que l'application se charge dans une SPA est appelé : Web App Gap

Web App Gap


Asset Downloads
Client Data
Server Response
Client Init
Paint
THE WEB APP GAP
0“
3“ - 7“

Web App Gap

" 57% des utilisateurs disent fermer une page si elle ne répond pas en moins de 3 secondes ”

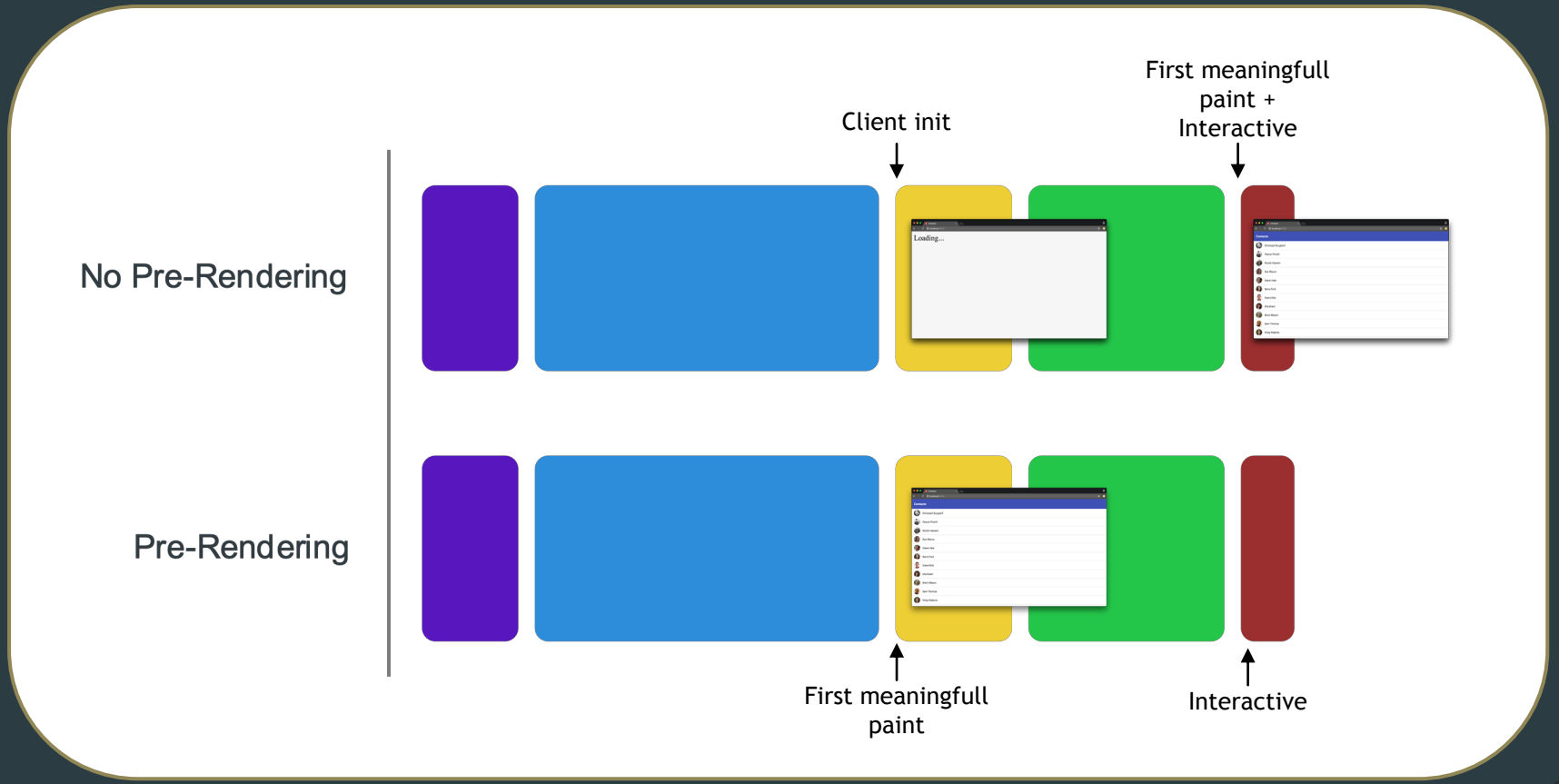
Web App Gap - compression

- La compression est la première étape d'une optimisation.
  - Elle permet de réduire la taille de l'application jusqu'à 3 fois.
- Mais elle ne peut jouer que sur deux parties du Web App Gap
  - Téléchargement des assets
  - Données des APIs

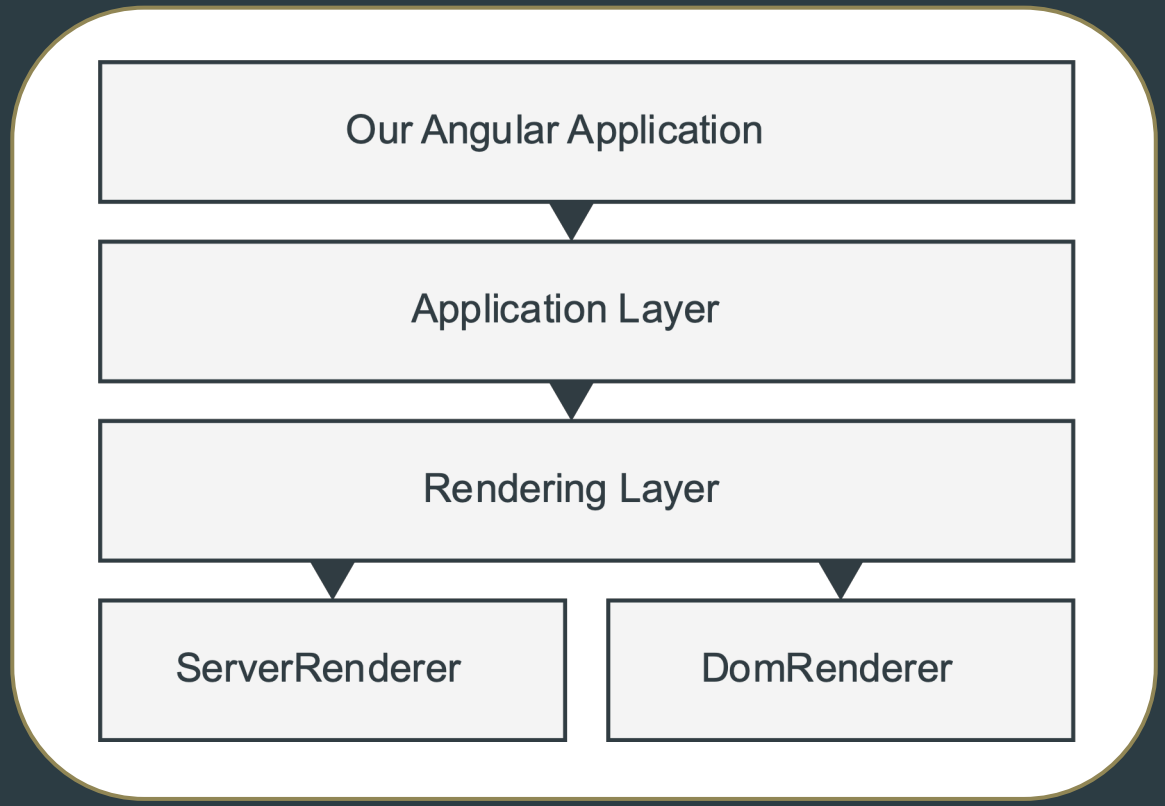
Web App Gap

- Deux métriques importantes
  - Premier rendu affichable
  - Interactivité

Web App Gap


Client init
First meaningfull paint + Interactive
No Pre-Rendering
Loading...
Pre-Rendering
First meaningfull paint
Interactive

Fonctionnement interne



Pas de DOM côté serveur

- Attention, aucun API du DOM n'existe côté serveur !

- Pour manipuler le DOM utilisez :
  - Les directives Angular (ngIf, ngClass…)
  - Le provider DOCUMENT
  - Le service Renderer2

Eviter les promises

- Angular Universal se base sur la référence des observables pour pouvoir déterminer si une page est complètement rendue.
- Il faut donc éviter de transformer ces observables en un autre type comme avec (toPromise())
- async/await sont donc également une fausse bonne idée avec Angular

Implémentation

https://angular.io/guide/universal

Infrastructure

- Serveurs compatibles
  - Les serveurs pouvant interpréter le JavaScript
    - **NodeJS**
    - ASP.NET Core
- Compatibilité en cours de développement
  - Par intégration du V8
    - Django (Python)
    - Go
    - PHP
    - Java

Outils nécessaires

- **Express**: création de serveurs web en NodeJS
- **Webpack**: compilation et bundeling du serveur

Fichiers impactés

- server.ts: nouveau serveur NodeJS
- app.module.ts: module client principal
- app.server.module.ts: nouveau module principal serveur
- main.server.ts: point d'entrée de la partie serveur
- tsconfig.server.ts: configuration typescript
- webpack.server.config.js: configuration Webpack

Ajouter Universal à un projet

- Angular CLI nous fournit maintenant l'outil pour ajouter facilement Universal
  - ng add @nguniversal/express-engine --clientProject aa-universal-workshop
- Lancez l'application en mode universal
  - npm run build:ssr && npm run serve:ssr
- Cela semble fonctionner, mais il y a une erreur dans la console :
  - Error: Cannot find module '../products/products.module.ngfactory'
  - En effet, il manque une dépendance pour que lazyloading soit pris en compte.
    - Pour cela il faut importer le module « ModuleMapLoaderModule » dans « app.server.module.ts »

angular.json

- Modification du outputPath de l'application principale afin de l'exporter dans le dossier « browser »
- Ajout du type de build « server » qui sera exportée dans le dossier « server »

package.json

- Nouveaux packages
  - @angular/platform-server: gestion de la plateforme « serveur » dans Angular
  - @nguniversal/express-engine: Universal Engine pour Express
  - @nguniversal/module-map-ngfactory-loader: gestion du lazyload
  - express: outil de création de server NodeJS
  - webpack-cli: outil de build
  - ts-loader: loader WebPack pour compatibilité avec TypeScript

server.ts

```
// Création d'une application Express
const app = express();

// Setup de l'engine Universal dans Express
const {AppServerModuleNgFactory, LAZY_MODULE_MAP} = require('./dist/server/main');
app.engine('html', ngExpressEngine({
    bootstrap: AppServerModuleNgFactory,
    providers: [
        provideModuleMap(LAZY_MODULE_MAP)
    ]
}));

// Mise a disposition des fichiers statiques (dossier browser)
app.get('*.*', express.static(DIST_FOLDER, {
    maxAge: '1y'
}));

// Utiliser l'engine Universal pour toutes les autres routes
app.get('*', (req, res) => {
    res.render('index', { req });
});
```

app.module.ts

- Ajout d'un appId afin de pouvoir créer un lien entre l'application CSR et l'application SSR
  - BrowserModule.withServerTransition({ appId: 'serverApp' })

app.server.module.ts

- Ce module importe l'application cliente, mais également « ServerModule ».
- Attention à bien mettre le ServerModule après l'AppModule car il écrase certains fonctionnements de base:
  - Http: il utilise la librairie http spécifique à NodeJS
  - Location: utilisé pour le routing
  - DOM: c'est Domino qui est utilisé pour la génération du DOM

main.server.ts

- L’application étant bootstrapée dans Express, ce fichier ne doit faire qu’exporter le module qui sera utilisé.
  - `export { AppServerModule } from './app/app.server.module';`

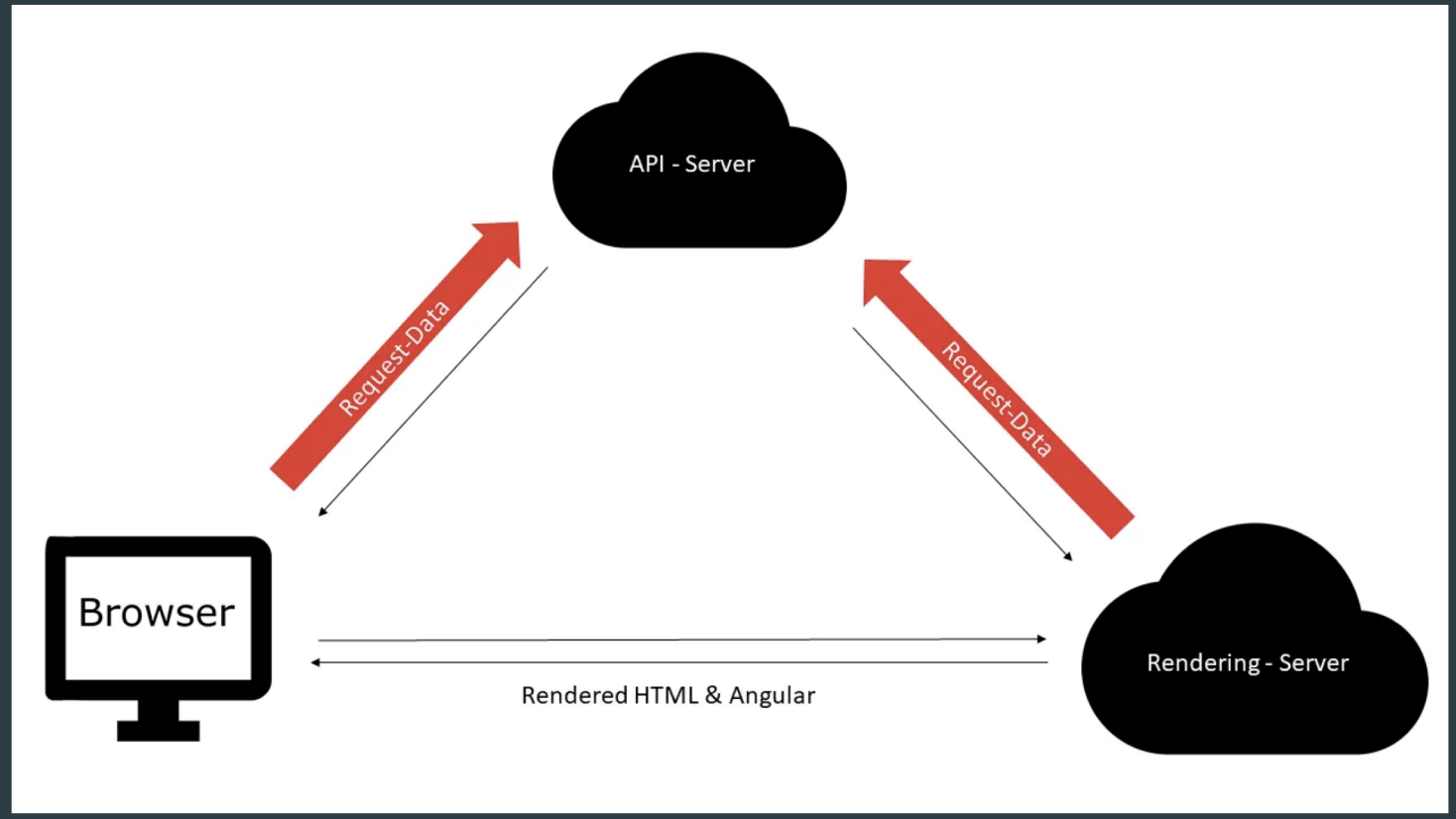
webpack.server.config.js

- Build le fichier « server.ts » ainsi que toutes ces dépendances
- Package le tout dans un seul fichier nommé « server.js »

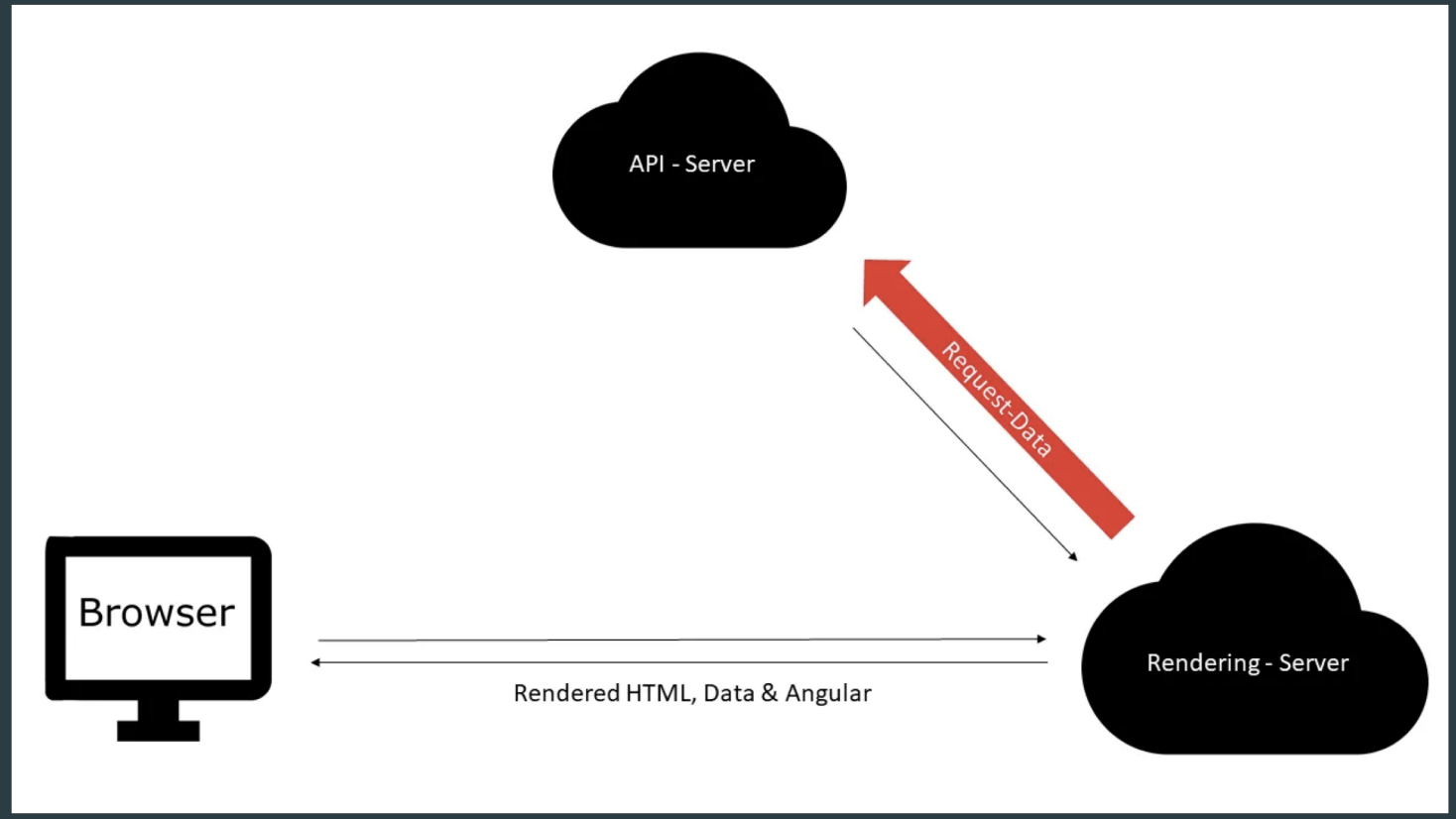
TransferState

Éviter les doubles appels API

Sans transfer state



Avec transfer state



main.ts

- L'état est écrit dans le DOM en tant qu'objet Json dans une balise <script>.
- Afin qu'Angular attende que l'état soit interprété par le navigateur avant de lancer l'application, il faut encapsuler le bootstraping « bootstrapModule » dans l'événement « DOMContentLoaded »

```
document.addEventListener('DOMContentLoaded', () => {
    platformBrowserDynamic().bootstrapModule(AppModule)
        .catch(err => console.log(err));
});
```

Les modules

- Ajout du module « BrowserTransferStateModule » dans le fichier « app.module.ts »
- Ajout du module « ServerTransferStateModule » dans le fichier « app.server.module.ts »

pizzas.service.ts

```
const PIZZAS_KEY = makeStateKey('pizzas');

getPizzas(): Observable<Pizza[]> {
    const found = this.transferState.hasKey(PIZZAS_KEY);
    if (found) {
        const res = of(this.transferState.get<Pizza[]>(PIZZAS_KEY, []));
        this.transferState.remove(PIZZAS_KEY);
        return res;
    } else {
        let resultToSerialize: Pizza[];
        this.transferState.onSerialize(PIZZAS_KEY, () => resultToSerialize);
        return this.http
            .get<Pizza[]>(`${environment.baseUrl}/pizzas`)
            .pipe(
                tap(result => resultToSerialize = result),
                catchError((error: any) => throwError(error.json()))
            );
    }
}
```

Universal avancé

Encore quelques petits trucs

ngExpressEngine vs AppServerModuleNgFactory

- ngExpressEngine = AppServerModule préconfiguré simple à utiliser
- AppServerModuleNgFactory = Customizable

```
app.engine('html', ngExpressEngine({
    bootstrap: AppServerModuleNgFactory,
    providers: [
        provideModuleMap(LAZY_MODULE_MAP)
    ]
}));
```

VS

```
const template = readFileSync(join(DIST_FOLDER, 'browser', 'index.html')).toString();
app.engine('html', (_, options, callback) => {
    renderModuleFactory(AppServerModuleNgFactory, {
        document: template,
        url: options.req.url,
        extraProviders: [
            provideModuleMap(LAZY_MODULE_MAP)
       ]
    }).then(html => {
        callback(null, html);
    });
});
```

Page 404

- Dans une SPA, il est commun de gérer la gestion de routes manquantes à l'aide d'une redirection ou d'une page générée côté serveur, mais en SSR il est plus commun de gérer cela via un code HTTP 404.
- Cela est également possible via Universal en injectant le provider « RESPONSE ».

```
@Component({
    selector: 'app-notfound',
    templateUrl: './notfound.component.html'
})
export class NotfoundComponent implements OnInit {
    constructor(
        @Inject(PLATFORM_ID) private platformId: Object,
        @Optional() @Inject(RESPONSE) private response: Response) {
    }
    ngOnInit() {
        if (!isPlatformBrowser(this.platformId)) {
            this.response.status(404);
        }
    }
}
```

Cache

- Sans cache

```
app.get('*', (req, res) => {
    res.render('index', { req });
});
```

- Avec cache

```
function render(req, res, url) {
    res.render('index', { req }, (err, html) => {
        cacheManager.set(url, html);
        res.status(200).send(html);
    });
}

app.get('*', (req, res) => {
    const url = req.originUrl;
    cacheManager.get(url).then((result) => {
        res.status(200).send(result);
    }, (error) => {
        render(req, res, url);
    });
});
```

SEO – Metadata

```
import { Title, Meta } from '@angular/platform-browser’;

// SEO
this.title.setTitle(data.title);
this.meta.addTag({name: 'description', content: data.longDescription});

// Twitter
this.meta.addTag({name: 'twitter:title', content: data.title});

// Facebook
this.meta.addTag({property: 'og:title', content: data.title});
```

- Attention à bien nettoyer vos metadata avant d’en rajouter

```
this.meta.removeTag('name="description"');
this.meta.removeTag('name="twitter:title"');
this.meta.removeTag('property="og:title"');
```

Dynamisation des paramètres de l'environnement

- Côté Express

```
const config = {
    provide: 'CONFIG',
    useValue: {
        baseServiceUrl: process.env.BaseServiceUrl,
        localesUrl: process.env.LocalesUrl,
        captchaKey: process.env.CaptchaKey
    }
};
```

```
app.engine('html', ngExpressEngine({
    bootstrap: AppServerModuleNgFactory,
    providers: [
        provideModuleMap(LAZY_MODULE_MAP),
        config
    ]
}));
```

Dynamisation des paramètres de l'environnement

- Dans AppModule

```
@Optional() @Inject('CONFIG') config
```

```
const CONFIG_KEY = makeStateKey<{ baseUrl: string }>('config');
```

```
let mainConfig: any;

if (this.isServer) {
    mainConfig = config;
    this.tstate.onSerialize(CONFIG_KEY, () => {
        return mainConfig;
    });
} else {
    mainConfig = this.tstate.get(CONFIG_KEY, null);
}

if (mainConfig) {
    if (mainConfig.baseServiceUrl) {
        environment.baseUrl = mainConfig.baseServiceUrl;
    }
}
```

Dynamisation des paramètres de l'environnement

- Pour tester si vous recevez bien le paramètre jusqu'au client

```
cd dist && BaseServiceUrl=http://test.com/api/ node server.js
```

Dynamisation des paramètres de l'environnement

- Pour faire cette configuration pour ngx-translate, il sera nécessaire d'utiliser https://github.com/gilsdav/configurable-translate-http-loader

```
export function createTranslateLoader(http: HttpClient) {
    return new ConfigurableTranslateHttpLoader(http, environment.locales, '.json');
}
```

```
if (mainConfig) {
    if (mainConfig.localesUrl) {
        environment.locales = mainConfig.localesUrl;
    }
}

ConfigurableTranslateHttpLoader.localesPathSubject.next(environment.locales);
```

Compatibilité localize-router

- Le localize-router n'est, de base, pas compatible avec le lazy-loading Universal
- Pour ce faire vous devrez ajouter ce provider dans votre AppServerModule https://github.com/gilsdav/universal-localize-module-loader

```
@NgModule({
    imports: [
        ...
    ],
    providers: [
        // Add universal-only providers here
        LazyUniversalModuleLoaderProvider
    ],
    bootstrap: [ AppComponent ],
})
export class AppServerModule {}
```

SEO – Outils

- Extensions Chrome
  - META SEO Inspector: https://www.omiod.com/meta-inspector.php
  - Open Graph Preview https://chrome.google.com/webstore/detail/open-graph-preview/ehaigphokkgebnmdiicabhjhddkaekgh
- Debuger Facebook : https://developers.facebook.com/tools/debug/

Références

- https://angular.io/guide/universal
- https://blog.thoughtram.io/announcements/2018/04/12/rxjs-master-class-and-courseware-updates.html
- https://blog.lysender.com/2018/07/angular-6-x-404-page-with-correct-header-using-angular-universal/
- https://malcoded.com/posts/angular-fundamentals-universal-server-side-rendering